Optimal speed scaling under arbitrary power functions

Lachlan L.H. Andrew
Centre for Advanced Internet Architectures
Swinburne University of Technology, Australia

Adam Wierman
Computer Science Department
California Institute of Technology

Ao Tang
School of Electrical and Computer Engineering
Cornell University

ABSTRACT

This paper investigates the performance of online dynamic speed scaling algorithms for the objective of minimizing a linear combination of energy and response time. We prove that (SRPT, $P^{-1}(n)$), which uses Shortest Remaining Processing Time (SRPT) scheduling and processes at speed such that the power used is equal to the queue length, is 2-competitive for a very wide class of power-speed tradeoff functions. Further, we prove that there exist tradeoff functions such that no algorithm can attain a competitive ratio less than 2.

1. INTRODUCTION

No longer is "faster" always "better". Today's computer system designs must balance energy consumption with traditional quality of service measures such as response time (a.k.a. sojourn time, flow time). There is a growing literature on power management, cf. [10, 11, 17] and references therein. One fundamental aspect of many power management designs is *dynamic speed scaling* [3, 7, 20, 22], which reduces the processing speed at times of low workload since processing more slowly uses less energy per operation. Speed scaling is now common practice in CPU designs [8, 9], and has been proposed in many other settings, e.g., switch fabrics [13], wireless networks [6] and TCP offload engines [14].

The growing practical importance of speed scaling designs has spurred analytic research into the design and analysis of algorithms for this problem. An algorithm for the *speed scaling problem* must make two decisions at any given time: (i) a *scheduling policy* must determine which job to service, and (ii) a *speed scaling scheme* must determine how fast to run the server. Let $(p, s(n))$ denote a speed scaling policy with scheduling policy p and speed $s(n)$ when there are n jobs in the system. In particular, we will be interested in (SRPT, $P^{-1}(n)$), which uses Shortest Remaining Processing Time first scheduling, and when the system has n jobs, runs at a speed $s(n)$ such that $P(s(n)) = n$, where $P(s)$ is the energy cost associated with running at speed s.

The analytic study of algorithms for the speed scaling problem began with Yao et al. [19]. Since then, three main objectives for balancing energy and quality of service have been considered: (i) minimize the total energy used in order to complete arriving jobs by their deadlines, e.g., [10, 19]; (ii) minimize the average response time of jobs given a set energy/heat budget, e.g., [5, 15, 16, 21]; and (iii) minimize a linear combination of the response time and energy usage, e.g., [2, 11, 18]. Note that this third objective has an interpretation in terms of design tradeoffs when it is possible to quantify how much reduction in response time is required to justify using an extra 1 joule of energy.

In this work, we focus on the third objective, minimizing a linear combination of energy and response time. Albers and Fujiwara [1] initiated the study of this objective. Like most theoretical work, they studied the case when the power associated with processing at speed s, $P(s)$, is defined as $P(s) = s^\alpha$, which is taken as a model of the power function of CMOS when α is around 2–3, cf. [11, 18]. They considered non-preemptive schedules, and showed that the best scheme for arbitrary job sizes must have a competitive ratio unbounded in the number of jobs in the schedule. For the special case of unit-work jobs, they provided both an $O\left(\left(\frac{3+\sqrt{5}}{2}\right)^\alpha\right)$ competitive batch algorithm, and an efficient offline algorithm to determine the optimal schedule.

Next, Bansal et al. [4] designed a different algorithm that is 4-competitive in the case of unit-sized jobs. Following this work, focus shifted back to the case of general job size distribution, and Lam et al. [12] proved that (SRPT, $P^{-1}(n)$) is $O(\alpha/\log(\alpha))$-competitive if preemption is allowed.

Finally, a big step forward occurred this year when Bansal, Cheng and Pruhs [2] tightened the competitive ratio substantially, and also provided a single competitive ratio valid for a very general class of functions $P(\cdot)$, rather than only the class s^α. Define a "regular" power scaling function as one which is

- differentiable on $[0, \infty)$
- strictly concave
- non-negative, and 0 at the origin.

Bansal, Cheng and Pruhs [2] first show that the scaling (SRPT, $P^{-1}(n+1)$) is 3-competitive for "regular" power functions, They then use this to show that a related speed scaling is $(3+\varepsilon)$-competitive for arbitrary non-negative and unbounded power functions, and arbitrary $\varepsilon > 0$.

The main contribution of our paper is to further tighten the competitive ratio. In particular, we prove that the scaling (SRPT, $P^{-1}(n)$) is 2-competitive when the power function is regular and a related scaling is $(2+\varepsilon)$-competitive under arbitrary (non-negative and unbounded) power functions. Further, we prove that this result is tight in two senses. First, for any power function, there is an instance of arriving jobs such that (SRPT, $P^{-1}(n)$) is a factor of arbitrarily close to 2 worse than optimal. Second, for any scaling algorithm, there is a power function such that the competitive ratio under that power function is arbitrarily close to 2.

2. PRELIMINARIES

In this paper we consider the joint problem of speed scaling and scheduling to minimizing a linear combination of response time and energy usage. Specifically, a problem instance consists of N jobs, with the ith job having release time (arrival time) r_i and size (work) y_i. The scheduler is online, and so is not aware of job i until time r_i and at that point learns the size y_i of the job. We allow the scheduler to be preemptive and assume that after preemption jobs may be restarted from the point they were interrupted without any overhead. At all points, the scheduler must decide (i) which job to serve and (ii) the processing speed. The speed is simply the rate at which work is completed, i.e., a job of size y_i served at speed s will complete in time y_i/s.

The power incurred from running at speed s is denoted by $P(s)$, and there is no penalty for switching from one speed to another. We will again focus on *regular* power functions. However, as shown in [2], it is possible for a system with any non-negative power function $\tilde{P}(\cdot)$ with unbounded support to emulate such a power function (provided the cost of switching speeds is negligible) in such a way as to increase the competitive ratio by no more than an arbitrarily small ε.

The objective that we consider is a linear combination of response time and energy usage. Let T_i be the response time of job i, the completion time minus the release time. Let s_t be the speed used at time t, then the total energy used is $E(I) = \int_I P(s_t)dt$. The cost of an instance I under a given algorithm A is then $C_A(I) = \sum_i T_i + \beta E(I)$. Without loss of generality, we set the unit of E such that $\beta = 1$.

The speed scaling algorithms that we study will choose the speed as a function of n, $s(n)$. We will focus on speed scaling algorithms defined by $P(s(n)) = n$. We will evaluate the performance of these algorithms by comparing them to the optimal (maybe offline) algorithm, OPT. In particular, we study the competitive ratio, defined as $\sup_I C_A(I)/C_{OPT}(I)$ where $C_{OPT}(I)$ is the optimal cost achievable on instance I.

3. RESULTS AND DISCUSSION

The (SRPT, $P^{-1}(n)$) algorithm has been suggested as a good speed scaling algorithm by a number of previous papers, cf. [1, 12]. It has previously been shown that a slight variation of this algorithm, (SRPT, $P^{-1}(n+1)$), is 3-competitive for regular power functions and leads to a $3+\varepsilon$ competitive algorithm in general [2]. Our main result is to tighten this competitive ratio as follows.

THEOREM 1. *For any regular power function P, (SRPT, $P^{-1}(n)$) has a competitive ratio of exactly 2.*

This theorem gives upper and lower bounds on the competitive ratio. The arguments for each are sketched below.

The proof of the upper bound uses similar techniques as were used in [2]. It uses an amortized local competitiveness argument with the potential function:

$$\Phi(t) = \int_0^\infty f(n^t(q))\,dq \tag{1}$$

where f is defined recursively by $f(0) = 0$, $f(i) = f(i-1) + \Delta(i)$, for some $\Delta(i)$, and $n^t(q) = \max(0, n_a^t(q) - n_o^t(q))$ with $n_a^t(q)$ and $n_o^t(q)$ the number of unfinished jobs at time t with remaining size at least q under the scheme under investigation and the optimal scheme, respectively. In particular, we will use the case when $\Delta(i) = 2P'(P^{-1}(i))$. The key to obtaining a tighter result than the one of [2] is the following.

LEMMA 2. *Let s_o and s_a be the speeds of the optimal scheme and (SRPT, $P^{-1}(n)$) at a given time, and let n_o and n_a be the numbers of unfinished jobs at that time. For any scheme s, any regular power function P and any Δ, if $n_o < n_a$ then either*

$$\text{both} \quad \tfrac{d}{dt}\Phi \le \Delta(n_a - n_o + 1)(-s_a + s_o) \tag{2a}$$
$$\text{and} \quad n_o \ge 1, \tag{2b}$$
$$\text{or} \quad \tfrac{d}{dt}\Phi \le \Delta(n_a - n_o)(-s_a + s_o). \tag{2c}$$

This differs from the corresponding result in [2] in the condition (2b). That ensures that the argument of Δ in (2) is always at most n_a, which gives the following.

LEMMA 3. *For $\Delta(i) = 2P'(P^{-1}(i))$, if $n_o < n_a$ then (2) implies*

$$\frac{d\Phi}{dt} \le 2(P(s_o) - n_a + n_o). \tag{3}$$

The remaining details of the upper bound on the competitive ratio is as used in [2].

The lower bound, that the competitive ratio of (SRPT, $P^{-1}(n)$) is at least 2, can be established as follows: Consider periodic unit-work arrivals at rate $q = s(n)$ for some n. The optimal schedule runs at rate q, and maintains a queue of at most one packet (the one in service), giving a cost per job of at most $(1 + P(q))/q$. In order to run at speed q, the schedule $P(s(n)) = n$ requires $n = P(q)$, giving a cost per job of $(P(q) + P(q))/q$. As we let the number of jobs that arrive grow large, the competitive ratio approaches

$$\frac{2P(q)}{1 + P(q)}.$$

Finally, as q becomes large the competitive ratio tends to 2 since $P(q)$ is unbounded.

This upper bound can then be used to extend the result to non-negative power functions using the same argument as in [2].

COROLLARY 4. *Let $\varepsilon > 0$. For any non-negative and unbounded $\tilde{P}$, there exists a P such that emulating (SRPT, $P^{-1}(n)$) yields a $(2+\varepsilon)$-competitive algorithm.*

A central part of the emulation involves avoiding speeds where P is not convex, instead emulating such speeds by switching between a higher and lower speed on the convex hull of $\tilde{P}$. This approach is very sensitive to the assumption that there is negligible cost incurred by changing speeds. Further work is required to investigate the impact of this very frequent switching.

The upper bound in Theorem 1 is tight, in two senses. The first sense is embodied in the theorem itself: for any power function, there exists an instance such that (SRPT, $P^{-1}(n)$) is a factor of 2 worse than optimal.

The second sense is that, for any speed scaling algorithm, there exists a power function such that the competitive ratio is arbitrarily close to 2.

THEOREM 5. *For any $\varepsilon > 0$ there is a regular power function P_ε such that for any speed scaling algorithm A, the competitive ratio of A is larger than $2 - \varepsilon$.*

The proof of this result is constructive and shows an instance and power function where a given speed scaling algorithm will be at least $(2-\varepsilon)$-competitive. Specifically, it uses $P(s) = s^\alpha$, all jobs of size 1, and two simple arrival patterns: periodic arrivals and batch arrivals. The argument shows that, when α is large, if $s(n)$ grows too slowly then a high backlog must be maintained to run fast enough to handle the periodic instance; conversely if $s(n)$ grows too quickly then it will run too fast in the case when no more arrivals occur (the batch arrival case).

Even though (SRPT, $P^{-1}(n)$) is 2-competitive and this is the best competitive ratio possible for any speed scaling algorithm given an arbitrary power function, if the power function is known, it may be possible to improve upon (SRPT, $P^{-1}(n)$). In the case that $P(s)$ has bounded derivative (but unbounded support), there exists a $k > 1$ such that (SRPT, $P^{-1}(kn)$) has competitive ratio is strictly less than 2. In particular, if $P(s)$ is linear, then the competitive ratio tends to 1 for large k.

We conjecture that, for regular functions, the scaling with the lowest possible competitive ratio will in general have the form (SRPT, $P^{-1}(kn+o(n))$) for some $k \geq 1$. In particular, under the optimal speeds, it can be shown that $P(s(n))$ grows at least as fast as n, but not as fast as $\omega(n)$. This is the subject of ongoing work.

4. CONCLUDING REMARKS

The results in this paper show that, for regular power functions, (SRPT, $P^{-1}(n)$) achieves a competitive ratio of 2; this is the best possible competitive ratio for the objective of minimizing a linear combination of response time and power, without further information about the shape of P being known.

Our current work on this topic is looking to extend this analysis in number of important directions. First, in many applications it is not possible to schedule using SRPT. However, even under different scheduling policies it is natural to expect that choosing $s(n)$ such that $P(s(n)) = n$ should work well. Second, we have proven that dynamic speed scaling using $P(s(n)) = n$ is 2-competitive, but how well does it compare to the case when an optimal static speed is chosen as a function of the workload? It may be possible to achieve nearly the same performance using a scheme that uses $s(0) = 0$ and for $n \geq 1$ uses $s(n) = q$, where q is a fixed speed optimally tuned to workload parameters. This scheme would be much more implementable in practice than a completely dynamic speed scaling design. To answer this question, we will need to perform an analysis of the speed scaling problem in a stochastic environment [18], rather than the worst-case framework considered in this paper.

5. REFERENCES

[1] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. In *Lecture Notes in Computer Science (STACS)*, volume 3884, pages 621–633, 2006.

[2] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proc. ACM-SIAM SODA*, 2009.

[3] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):1–39, Mar. 2007.

[4] N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow times. In *Proc. ACM-SIAM SODA*, pages 805–813, 2007.

[5] D. P. Bunde. Power-aware scheduling for makespan and flow. In *Proc. ACM Symp. Parallel Alg. and Arch.*, 2006.

[6] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. In *Proc. ACM SIGCOMM*, pages 135–146, Seattle, WA, Aug. 2008.

[7] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proc. ISLPED*, page 6, 2007.

[8] IBM PowerPC. http://www-03.ibm.com/technology/power/powerpc.html.

[9] Intel Xscale. www.intel.com/design/intelxscale.

[10] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[11] S. Kaxiras and M. Martonosi. *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool, 2008.

[12] T.-W. Lam, L.-K. Lee, I. K. K. To, and P. W. H. Wong. Speed scaling functions for flow time scheduling based on active job count. In *Proc. European Symposium on Algorithms*, 2009.

[13] L. Mastroleon, D. O'Neill, B. Yolken, and N. Bambos. Power aware management of packet switches. In *Proc. High-Perf. Interconn.*, 2007.

[14] S. Narendra et al. Ultra-low voltage circuits and processor in 180 nm to 90 nm technologies with a swapped-body biasing technique. In *Proc. IEEE Int. Solid-State Circuits Conf*, page 8.4, 2004.

[15] K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. In *Scandinavian Worksh. Alg. Theory*, 2004.

[16] K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. In *Proc. Worksh. Approx. Online Alg.*, 2005.

[17] O. S. Unsal and I. Koren. System-level power-aware deisgn techniques in real-time systems. *Proc. IEEE*, 91(7):1055–1069, 2003.

[18] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *Proc. IEEE INFOCOM*, 2009.

[19] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 374–382, 1995.

[20] L. Yuan and G. Qu. Analysis of energy reduction on dynamic voltage scaling-enabled systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 24(12):1827–1837, Dec. 2005.

[21] S. Zhang and K. S. Catha. Approximation algorithm for the temperature-aware scheduling problem. In *Proc. IEEE Int. Conf. Comp. Aided Design*, pages 281–288, Nov. 2007.

[22] Y. Zhu and F. Mueller. Feedback EDF scheduling of real-time tasks exploiting dynamic voltage scaling. *Real Time Systems*, 31:33–63, Dec. 2005.